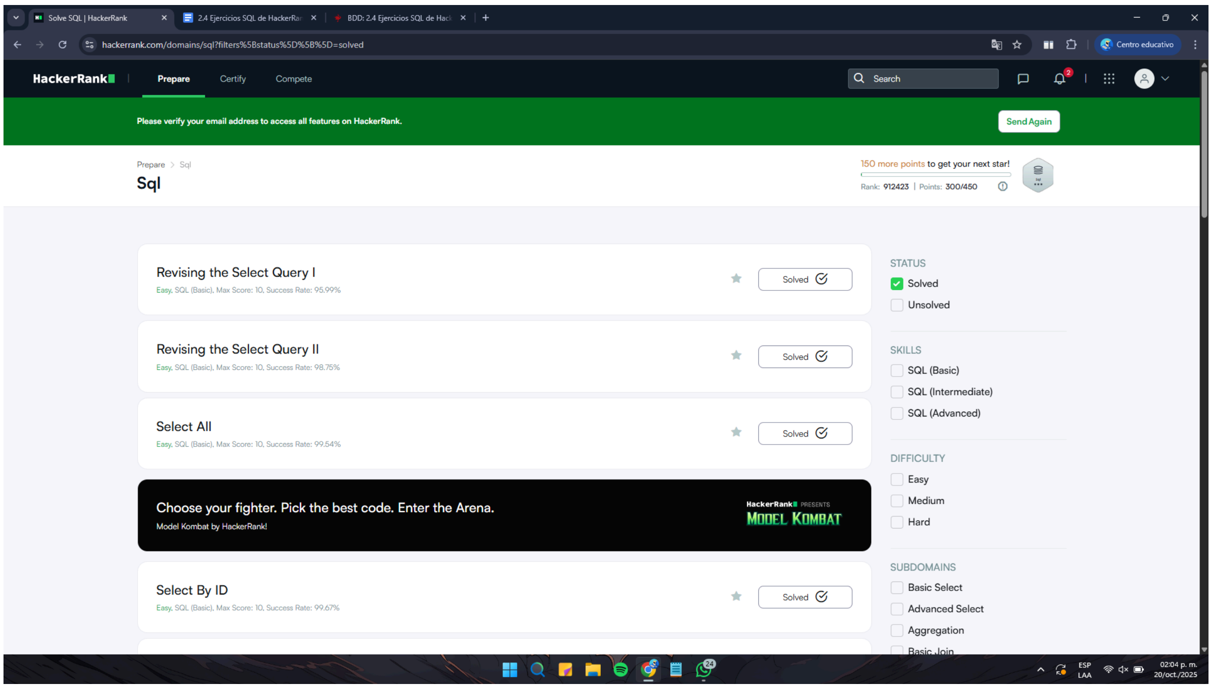
2.4 Ejercicios SQL de HackerRank

Jennifer Resendiz Isidro


Solve SQL | HackerRank
2.4 Ejercicios SQL de HackerRan
BDD: 2.4 Ejercicios SQL de Hack
hackerrank.com/domains/sql?filters%5Bstatus%5D%5B%5D=solved
HackerRank
Prepare
Certify
Compete
Search
Centro educativo
Send Again

Please verify your email address to access all features on HackerRank.

Prepare > Sql

Sql

150 more points to get your next star!
Rank: 912423 | Points: 300/450

Revising the Select Query I
Easy, SQL (Basic), Max Score: 10, Success Rate: 95.99% — Solved

Revising the Select Query II
Easy, SQL (Basic), Max Score: 10, Success Rate: 98.75% — Solved

Select All
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.54% — Solved

Choose your fighter. Pick the best code. Enter the Arena.
Model Kombat by HackerRank!

Select By ID
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.67% — Solved

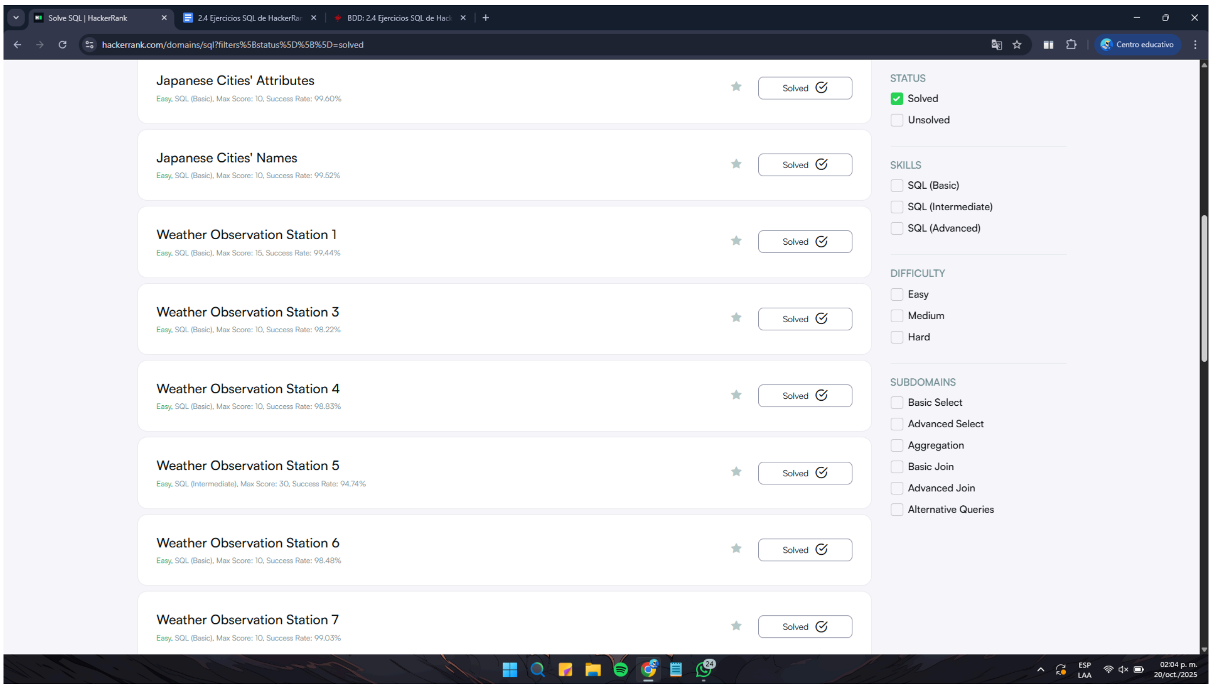
STATUS
- [✓ Solved]
- Unsolved

SKILLS
- SQL (Basic)
- SQL (Intermediate)
- SQL (Advanced)

DIFFICULTY
- Easy
- Medium
- Hard

SUBDOMAINS
- Basic Select
- Advanced Select
- Aggregation
- Basic Join

ESP LAA
02:04 p. m.
20/oct./2025


Solve SQL | HackerRank
2.4 Ejercicios SQL de HackerRan
BDD: 2.4 Ejercicios SQL de Hack
hackerrank.com/domains/sql?filters%5Bstatus%5D%5B%5D=solved
Centro educativo

Japanese Cities' Attributes
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.60% — Solved

Japanese Cities' Names
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.52% — Solved

Weather Observation Station 1
Easy, SQL (Basic), Max Score: 15, Success Rate: 99.44% — Solved

Weather Observation Station 3
Easy, SQL (Basic), Max Score: 10, Success Rate: 98.22% — Solved

Weather Observation Station 4
Easy, SQL (Basic), Max Score: 10, Success Rate: 98.83% — Solved

Weather Observation Station 5
Easy, SQL (Intermediate), Max Score: 30, Success Rate: 94.74% — Solved

Weather Observation Station 6
Easy, SQL (Basic), Max Score: 10, Success Rate: 98.48% — Solved

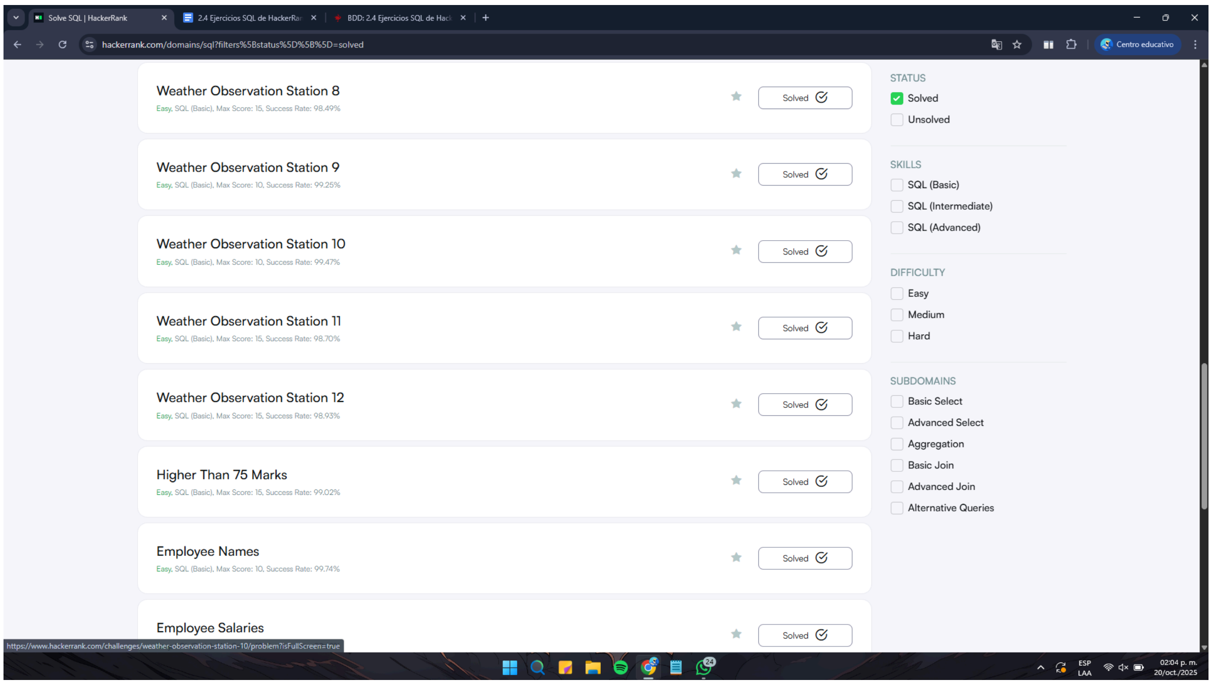
Weather Observation Station 7
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.03% — Solved

STATUS
- [✓ Solved]
- Unsolved

SKILLS
- SQL (Basic)
- SQL (Intermediate)
- SQL (Advanced)

DIFFICULTY
- Easy
- Medium
- Hard

SUBDOMAINS
- Basic Select
- Advanced Select
- Aggregation
- Basic Join
- Advanced Join
- Alternative Queries



Weather Observation Station 8
Easy, SQL (Basic), Max Score: 15, Success Rate: 98.49% — Solved

Weather Observation Station 9
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.25% — Solved

Weather Observation Station 10
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.47% — Solved

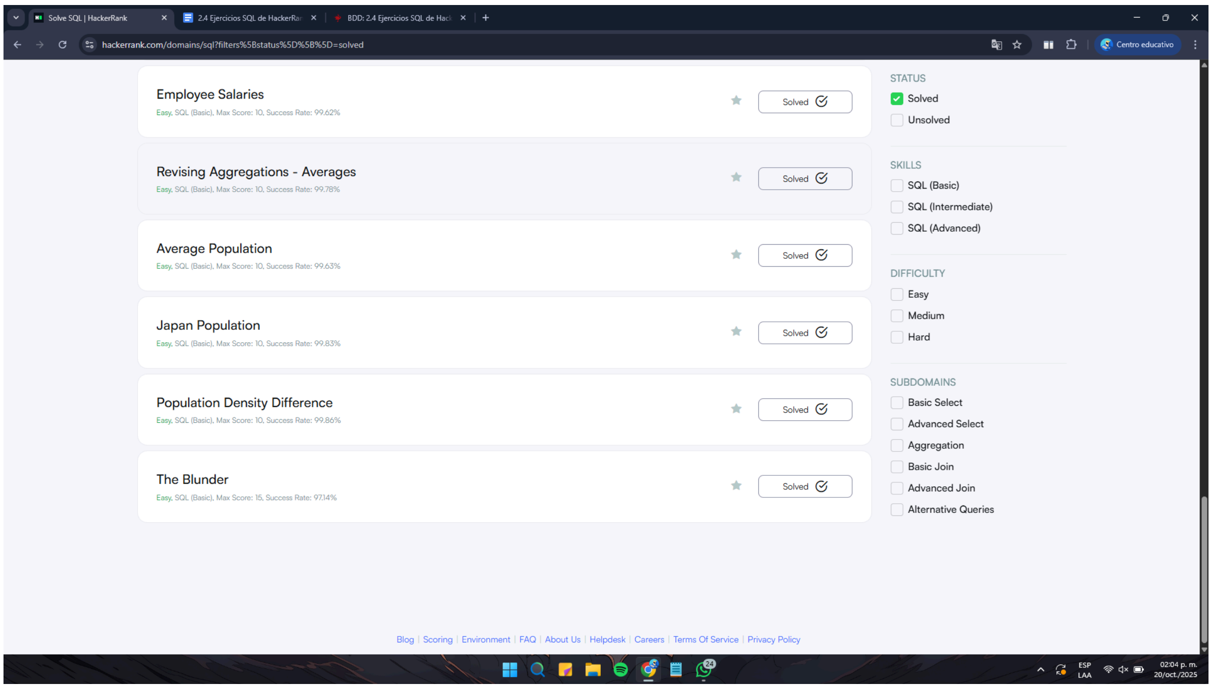
Weather Observation Station 11
Easy, SQL (Basic), Max Score: 15, Success Rate: 98.70% — Solved

Weather Observation Station 12
Easy, SQL (Basic), Max Score: 15, Success Rate: 98.93% — Solved

Higher Than 75 Marks
Easy, SQL (Basic), Max Score: 15, Success Rate: 99.02% — Solved

Employee Names
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.74% — Solved

Employee Salaries — Solved


Solve SQL | HackerRank
2.4 Ejercicios SQL de HackerRan
BDD: 2.4 Ejercicios SQL de Hack
hackerrank.com/domains/sql?filters%5Bstatus%5D%5B%5D=solved
Centro educativo
STATUS
Solved
Unsolved
SKILLS
SQL (Basic)
SQL (Intermediate)
SQL (Advanced)
DIFFICULTY
Easy
Medium
Hard
SUBDOMAINS
Basic Select
Advanced Select
Aggregation
Basic Join
Advanced Join
Alternative Queries
Blog | Scoring | Environment | FAQ | About Us | Helpdesk | Careers | Terms Of Service | Privacy Policy

Employee Salaries
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.62% — Solved

Revising Aggregations - Averages
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.78% — Solved

Average Population
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.63% — Solved

Japan Population
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.83% — Solved

Population Density Difference
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.86% — Solved

The Blunder
Easy, SQL (Basic), Max Score: 15, Success Rate: 97.14% — Solved

HACKER RANK---

Query all columns for all American cities in the CITY table with populations larger than 100000. The CountryCode for America is USA.

The CITY table is described as follows:

https://s3.amazonaws.com/hr-challenge-images/8137/1449729804-f21d187d0f-CITY.jpg

respuesta:

```
SELECT * FROM CITY WHERE POPULATION > 100000 AND COUNTRYCODE = 'USA';
```

Query the NAME field for all American cities in the CITY table with populations larger than 120000. The CountryCode for America is USA.

The CITY table is described as follows:

https://s3.amazonaws.com/hr-challenge-images/8137/1449729804-f21d187d0f-CITY.jpg

respuesta:

```
SELECT NAME FROM CITY WHERE POPULATION > 120000 AND COUNTRYCODE = 'USA';
```

Query all columns (attributes) for every row in the CITY table.

The CITY table is described as follows:

https://s3.amazonaws.com/hr-challenge-images/8137/1449729804-f21d187d0f-CITY.jpg

```
SELECT * FROM CITY
```

Query all columns for a city in CITY with the ID 1661.

The CITY table is described as follows:

Respuesta:

```
SELECT * FROM CITY WHERE ID = 1661
```

Query a list of CITY and STATE from the STATION table.

The STATION table is described as follows:

https://s3.amazonaws.com/hr-challenge-images/9336/1449345840-5f0a551030-Station.jpg

```
SELECT CITY, STATE FROM STATION;
```

Query a list of CITY names from STATION for cities that have an even ID number. Print the results in any order, but exclude duplicates from the answer.

The STATION table is described as follows:

```
SELECT DISTINCT CITY FROM STATION WHERE MOD(ID,2)=0;
```

Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.

The STATION table is described as follows:

```
SELECT count(city)- count(distinct city) as difference from station;
```

Query the two cities in STATION with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.

The STATION table is described as follows:

```
select CITY, length(CITY) from STATION order by length(CITY), CITY limit 1;

select CITY, length(CITY) from STATION order by length(CITY) desc, CITY limit 1;
```

Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates.

Input Format

```
SELECT DISTINCT CITY FROM STATION WHERE lower(substr(CITY,1,1)) in ('a','e','i','o','u') ;
```

Query the list of CITY names ending with vowels (a, e, i, o, u) from STATION. Your result cannot contain duplicates.

Input Format

```
select distinct CITY from STATION where right(city,1) in ('a', 'e', 'i', 'o', 'u');
```

Query the list of CITY names from STATION which have vowels (i.e., a, e, i, o, and u) as both their first and last characters. Your result cannot contain duplicates.

```
SELECT DISTINCT CITY FROM STATION WHERE LOWER(RIGHT(CITY, 1)) IN ('a', 'e', 'i', 'o', 'u') AND LOWER(SUBSTR(CITY, 1, 1)) IN ('a', 'e', 'i', 'o', 'u');
```

Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.

```
SELECT DISTINCT CITY FROM STATION WHERE NOT lower(substr(CITY,1,1)) in ('a','e','i','o','u') ;
```

Query the list of CITY names from STATION that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates.

```
SELECT DISTINCT CITY

FROM STATION

WHERE LOWER(LEFT(CITY, 1)) NOT IN ('a', 'e', 'i', 'o', 'u')

   OR LOWER(RIGHT(CITY, 1)) NOT IN ('a', 'e', 'i', 'o', 'u');
```

Query the list of CITY names from STATION that do not start with vowels and do not end with vowels. Your result cannot contain duplicates.

```
SELECT DISTINCT CITY

FROM STATION

WHERE LOWER(LEFT(CITY, 1)) NOT IN ('a', 'e', 'i', 'o', 'u')

  AND LOWER(RIGHT(CITY, 1)) NOT IN ('a', 'e', 'i', 'o', 'u');
```

Query the Name of any student in STUDENTS who scored higher than Marks. Order your output by the last three characters of each name. If two or more students both have names ending in the same last three characters (i.e.: Bobby, Robby, etc.), secondary sort them by ascending ID.

```
SELECT NAME FROM STUDENTS WHERE MARKS > 75 ORDER BY SUBSTR(NAME, -3), ID;
```

Write a query that prints a list of employee names (i.e.: the name attribute) from the Employee table in alphabetical order.

```
SELECT NAME FROM EMPLOYEE ORDER BY NAME;
```

Write a query that prints a list of employee names (i.e.: the name attribute) for employees in Employee having a salary greater than per month who have been employees for less than months. Sort your result by ascending employee_id.

```
SELECT name FROM Employee WHERE Salary > 2000 AND months < 10 ORDER BY employee_id ASC;
```

Query the average population of all cities in CITY where District is California.

```
SELECT AVG(POPULATION) FROM CITY WHERE DISTRICT = 'California';
```

Query the average population for all cities in CITY, rounded down to the nearest integer.

SELECT FLOOR(AVG(POPULATION)) FROM CITY;

Query the sum of the populations for all Japanese cities in CITY. The COUNTRYCODE for Japan is JPN.

SELECT SUM(POPULATION) FROM CITY WHERE COUNTRYCODE = 'JPN';

Query the difference between the maximum and minimum populations in CITY.

SELECT MAX(POPULATION) - MIN(POPULATION) FROM CITY;

Samantha was tasked with calculating the average monthly salaries for all employees in the EMPLOYEES table, but did not realize her keyboard's key was broken until after completing the calculation. She wants your help finding the difference between her miscalculation (using salaries with any zeros removed), and the actual average salary.

Write a query calculating the amount of error (i.e.: average monthly salaries), and round it up to the next integer.

SELECT CEIL(AVG(SALARY) - AVG(REPLACE(SALARY, '0', ''))) FROM EMPLOYEES;